Decentralized Search over Personal Online Datastores: Architecture and Performance Evaluation

Mohamed Ragab[1], Yury Savateev[1], Helen Oliver[2], Thanassis Tiropanis[1], Alexandra Poulovassilis[2], Adriane Chapman[1], Ruben Taelman[3], and George Roussos[2]

[1] School of Electronics and Computer Science, University of Southampton, UK
ragab.mohamed, y.savateev, t.tiropanis, adriane.chapman@soton.ac.uk
[2] School of Computing and Mathematical Sciences, Birkbeck, University of London, UK
h.oliver, a.poulovassilis, g.roussos@bbk.ac.uk
[3] IDLab, Department of Electronics and Information Systems, Ghent University – imec, Ghent, Belgium
ruben.taelman@ugent.be

Abstract. Data privacy and sovereignty are open challenges in today's Web, which the *Solid* [4] ecosystem aims to meet by providing personal online datastores (pods) where individuals can control access to their data. Solid allows developers to deploy applications with access to data stored in pods, subject to users' permission. For the decentralised Web to succeed, the problem of search over pods with varying access permissions must be solved. The ESPRESSO framework takes the first step in exploring such a search architecture, enabling large-scale keyword search across Solid pods with varying access rights. This paper provides a comprehensive experimental evaluation of the performance and scalability of decentralised keyword search across pods on the current ESPRESSO prototype. The experiments specifically investigate how controllable experimental parameters influence search performance across a range of decentralised settings. This includes examining the impact of different text dataset sizes (0.5MB to 50MB per pod, divided into 1 to 10,000 files), different access control levels (10%, 25%, 50%, or 100% file access), and a range of configurations for Solid servers and pods (from 1 to 100 pods across 1 to 50 servers). The experimental results confirm the feasibility of deploying a decentralised search system to conduct keyword search at scale in a decentralised environment.

Keywords: Web Re-decentralisation · Decentralised Search · Personal Online Data Stores (pods) · Solid Framework.

1 Introduction

The current state of the Web witnesses user-generated data being kept within centralised data silos, monopolised by a few large corporations [1]. This centralisation of the Web poses significant risks to privacy and user autonomy [10] and

[4] https://solidproject.org

slows down data-driven innovation. Indeed, this centralisation prevents a vast amount of data from being available for search because certain types of data, especially personal and sensitive information, are too confidential for public access, limiting traditional search engines from indexing and making such data available, thereby limiting the breadth of searchable content.Web users end up with neither control over their data nor privacy [4], and developers have to build more data silos of their own in order to reach enough users.

With the overarching objective of reinstating user control and data governance, a number of initiatives have emerged [4,10,17] that strive to decentralise the World Wide Web, aiming to distribute the control and ownership of data and the underlying infrastructure [4]. The decentralised paradigm aims to empower individuals to manage which third parties have access to their data and for what purposes. Decentralisation aims to foster data-driven advancements such as enhanced data sharing and synchronisation among different applications [7]. Most prominent amongst these innovations, and exemplifying the concept of Web decentralisation, is the `Solid` technology suite [5], which aims to empower people with direct control of their data [17]. This is facilitated by decoupling data from applications and enabling users to curate their data within personal online data repositories, referred to as `pods`, requiring third parties to get pod owners' consent to access their data.

Web decentralisation holds high potential for creating a more equitable and transparent online environment [4]. However, there are significant challenges, particularly in search and query processing over online decentralised data stores. Most modern search engines only offer centralised search services, while the current decentralised search utilities across Solid applications [12] and other distributed, federated, or Linked data query processing systems [19] do not yet provide adequate solutions for distributed search over resources where different users, applications and search entities can have different access rights [16,20] (further details in Section 2).

In response to this research gap, our previous works [16,20] have proposed a vision and architecture of *Efficient Search over Personal Repositories - Secure and Sovereign* (*ESPRESSO* [6]). ESPRESSO is a framework that aims to enable large-scale search across Solid pods while respecting individuals' data sovereignty and differing access rights. The work in [16,20] focused on the challenges and design principles of a decentralised search system and provided an overview of an architecture that aims to enable efficient decentralised Web search over Solid pods. It also presented the implementation of the first ESPRESSO prototype and preliminary proof-of-concept experiments.

In this paper, we aim to extend that work and provide a comprehensive experimental evaluation of the performance and scalability of decentralised search across Solid pods. We present and discuss the results of testing and validating the second prototype of the ESPRESSO system with larger datasets, a variety of data distributions, and different Solid server and pod setups.

[5] Solid is a set of specifications that can have several implementations, e.g., *Digita* `https://www.digita.ai/` and *Bluesky* `https://blueskyweb.xyz/`

[6] `https://espressoproject.org`

The contributions of this paper are as follows: (1) Validating the viability of the ESPRESSO architecture for undertaking scalable decentralised keyword-based search operations over personal online data stores (pods) distributed across several Solid servers. (2) Performance and scalability evaluation via comprehensive experiments, specifying a set of controllable experimental parameters that directly impact decentralised search performance. Specifically, we test the impact of the number of solid servers, the number of pods they allocate, various scales and distributions of datasets, the number of files the datasets are split into, and varying percentages of access control permissions. (3) Extension of the ESPRESSO system architecture, including optimisations in the keyword indexing structures and the search algorithm proposed in [16]. (4) Identification of open challenges of decentralised search over Solid pods.

2 Related Work

The decentralised search problem has been tackled from various perspectives and research areas, including distributed databases (DBs), Peer-to-peer (P2P) search and query routing, and SPARQL distributed querying and link-following. Significant research in distributed DB systems has explored distributed search methods for querying federated databases across various organisations with varying levels of autonomy [10]. Distributed indexing techniques have also been proposed to support search across multiple databases [6]. However, most methods, excluding some prototypes like [11], presume access to query endpoints, indexes, and result caching, which may not always be available in decentralised settings. P2P data management systems and query routing have been studied for decentralised search [15]. Protocols like IPFS employ *distributed hash tables* (DHTs) for keyword-based search [2]. Yet, these methods do not address varying access controls over data resources, query endpoints, and distributed indexes. Last but not least, decentralised SPARQL querying in Linked Data settings would require establishing and maintaining endpoint metadata relating to every search entity, implementing access controls for selecting data resources, and enforcing caching control constraints during SPARQL *link-following* [19,9]. Meeting these prerequisites would result in significant increases in storage requirements, network utilisation, and computational needs.

Hence, the methodologies essential for conducting data search operations across decentralised Solid pods extend beyond the existing literature. This is because *access privileges* to pod data can differ among diverse search entities, and *caching limitations* might impose constraints on the dissemination of search outcomes through the network [20]. Recent research focuses on querying RDF data in Solid pods from a decentralised *Knowledge Graphs* (KGs) viewpoint, using the *Link Traversal Query Processing* (LTQP) approach [19,22], but still lacks emphasis on access control [22] or decentralised indexing [19]. To this end, the ESPRESSO project [16,20] marks an initial step in enabling large-scale decentralised keyword search over Solid pods, prioritizing data sovereignty and adherence to user access controls by using decentralised indexes over pod contents. We focus initially on keyword-based search; a vital preliminary stage to comprehend the demands and performance factors necessary for advanced struc-

tured distributed queries (e.g. SPARQL) or decentralised keyword search over structured/semi-structured data in personal datastores [5].

3 Preliminaries & ESPRESSO Framework

3.1 Solid Framework

As described in the Introduction, one of the most prominent decentralised technologies is Solid [12,17]. The foundation of the Solid ecosystem comprises three essential building blocks that enable authentication and access control to pods:

- **LDP**: Solid incorporates elements of the W3C *Linked Data Platform* (LDP [7]) recommendation to enable read/write access to pod-stored data resources (e.g., text, RDF, etc.) with special provisions for managing Linked Data.
- **WebIDs & Solid OIDC** [8] for identification and authentication. These standards connect agents to decentralised identifiers containing information such as trusted identity providers (*WebID specifications* [9]). This enables authentication between resource and authorisation servers without pre-existing trust relationships.
- **Web Access Control** [10] for regulating information sharing within the pod. This is a decentralised, cross-domain solution that authorises requests using Linked Data-expressed *Access Control Lists* (ACLs). It uses IRIs to identify both agents and resources. The ACLs can be tailored for individual resources or inherited from a parent container.

3.2 ESPRESSO Framework Overview

Figure 1 shows the architecture of ESPRESSO along with the core components that enable decentralised keyword search across Solid pods. Each of the following components is installed alongside each Solid server in the network: **(1) Pod Indexing App** – ***Brewmaster*** creates and maintains local indexes for the text files inside the pod, which include information about the files' access control. Also, the Brewmaster indexing app maintains relevant information about the addresses of these local indexes, plus metadata for search optimisation and filtering, in a *MetaIndex* file in a dedicated *ESPRESSO*

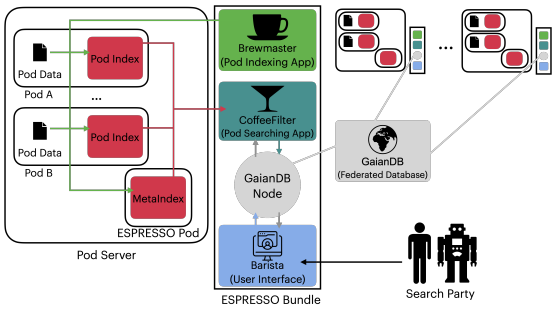


Fig. 1: *ESPRESSO* framework architecture.

[7] https://www.w3.org/TR/ldp/
[8] https://solidproject.org/TR/oidc
[9] https://www.w3.org/2005/Incubator/webid/spec/identity/
[10] https://solid.github.io/web-access-control-spec/

Pod on the Solid server. More details about the pod index structure and how it has evolved from the one in [16] are given in Section 3.3.

(2) User Interface – ***Barista*** serves as the user interface application that facilitates end-user search operations. It takes as input a keyword-based query and the user's WebID, and subsequently presents the ranked search results to the user.

(3) Overlay network: To propagate and route the user query across Solid servers, the ESPRESSO system utilises an overlay network of federated database nodes that connect the Solid servers, and returns aggregated search results retrieved from the Solid servers' pods to the user. Naturally, only results to which the searcher has access are returned. The current ESPRESSO system prototype uses a custom build of the GaianDB [11] [3] overlay network. A user's query can be initiated from any of the GaianDB nodes (connected to the Solid servers) and is automatically propagated to the other nodes in the network [3]. Thus, users can access all relevant data distributed across pods in different Solid servers through GaianDB nodes, according to their access rights.

(4) Pod Searching App – ***CoffeeFilter*** conducts local search operations on the pods of every Solid server on which it is installed. Upon receiving a query from the federated DB node, the CoffeeFilter Pod Searching App accesses the MetaIndex to retrieve the addresses of the local pod indexes. The CoffeeFilter Pod Searching App then performs a search against the relevant pod indexes and sends the results back to the federated DB node. The details of the search operation process are described in Section 3.3.

The *ESPRESSO Bundle* of the above components (shown in Figure 1) is available as an open-source software suite [12]. The Pod Indexer App (Brewmaster) needs to get access to the individual pod's content from the pod owners. Then, it can index the text files. The Pod Searching App (CoffeeFilter) needs access to the resulting indexes to search them.

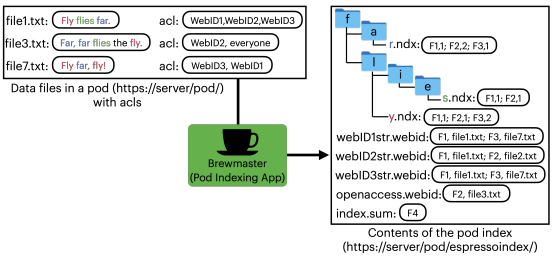


Fig. 2: Pod index structure. On the left, are pod files with corresponding ACLs. On the right, are generated pod indexing files.

3.3 Indexing & Search over Pods

In the preliminary investigation of implementing an inverted index for pods' textual data, we created a naive local index for a pod's text files which is described in [16]. The Search App must download this naive index for every query, lengthening search times and exposing data beyond the requirements of the query. Thus, for this paper, we developed an enhanced indexing scheme that reduces index

[11] GaianDB: `https://github.com/gaiandb/gaiandb`
[12] ESPRESSO Search System: `https://github.com/espressogroup/ESPRESSO`

size and increases emphasis on privacy, considering different users' and applications' access rights [23]. It considers the Solid server-side search processing limitations [7] and builds the indexing files to be accessed according to Solid *LDP* principles via simple HTTP GET requests [19]. It pays more attention to optimizing the size and the number of indexing files required for efficient search while respecting users' data sovereignty. Moreover, this approach prevents unnecessary data exposure beyond query necessities and access permissions, thereby strengthening the privacy of the system. Below, we describe how the new indexing scheme allows Solid applications and users to search accessible text data across pods.

Pod Indexing. Figure 2 shows how the Pod Indexing App (Brewmaster) creates local indexes based on the new indexing scheme.

Picture a Solid server at the address `https://server/` hosting a pod at the address `https://server/pod/`. Pod Indexing App (Brewmaster) assigns each file in the pod a short *fileID* according to a simple naming convention (e.g., *F1, F2, F3,...*) and creates an index directory `espressoindex`. This URL is stored in the MetaIndex in the server-level *ESPRESSO Pod*. In the pod index's directory, we organise the indexing files in subdirectories with a *trie*-like structure (see [14]) for faster access.

For each keyword, e.g., '*fly*', in the indexed text files, the Pod Indexing App (Brewmaster) creates a corresponding file in the index directory at the address `https://server/pod/espressoindex/f/l/y.ndx`. This file contains the fileIDs of each file containing the word '*fly*' and the number of times it appears in each. For each WebID that has read access to some of the indexed files e.g., *WebID1*, the Pod Indexing App creates a `webID1str.webid` file listing the short fileIDs with their full names (e.g. `file1.txt`). Publicly accessible files are listed, by fileID and filename, in the file `openaccess.webid`. Last, information required for updating the pod index is compiled into a file `index.sum`. The improved indexing scheme speeds up retrieval of index files by generating small, inverted micro-indexes for each keyword in text files; and enables access control with a list of WebIDs authorised to access files containing each keyword.

Search on Pods Indexes. Suppose the Pod Searching App (CoffeeFilter) receives a search request for the word *'fly'* coming from some *WebID*. First, it gets the *MetaIndex* from the *ESPRESSO Pod* and obtains the list of all the `podindex` URLs. Then, for each URL, it combines the *.ndx* and *.webid* data, returning a list of relevant URLs accessible to the search party's WebID, with the corresponding keyword frequencies. Finally, the Pod Searching App combines the results for all the pods on the server. The results are sent back to the overlay network.

4 Experiments

This section experimentally evaluates the performance and scalability of decentralised keyword search in ESPRESSO over Solid pods. The goal is to assess the viability of the decentralised search in our ESPRESSO prototype by exploring its performance under typical conditions. In particular, we investigate a range of

factors characterizing data distribution across Solid servers, such as the number of servers, the number of pods, the number and size of the files, and how many of those files the search party has access to.

4.1 Experimental Environment & Setup

Environment: Our experimentation, advancing beyond prior research which used a single Solid server [18,22], opted for an initial cluster of 50 virtual machines (VMs) to simulate a multi-organisation environment, such as a network of general health practitioners or other professionals in a metropolitan area, each using a Solid pod server where each of their clients can securely store their data in individual pods. This setup not only facilitates multi-server scenarios but also aids in empowering query propagation and routing among multiple Solid servers. Each of those VMs is equipped with the *Red Hat Enterprise Linux* 8.7 operating system and runs on 2.4GHz processors and it has 8GB of RAM. Additionally, each VM possesses a high-speed storage drive with a capacity of 125GB; and runs a single instance of *Community Solid Server* (*V*.6.0) with a file-based storage backend. The VMs are physically allocated from the data centre at Southampton University. The *CoffeeFilter* Pod Searching App leverages the *Node-js Axios* library for performing search requests over Solid Servers. We also used a custom build of GaianDB version 2.1.8 (i.e. extended with our *Solid-to-GaianDB* connector). The Solid-to-GaianDB connector uses *Logical Tables* [3] to create an abstract federation layer within the GaianDB network, integrating data from different Solid servers.
Dataset: We conducted our keyword search over a machine translation workshop dataset [13] [21]. The dataset mainly comprises text from the *News Crawl Corpus* [14] and we extracted the text in English only, amounting to 14GB of textual source data.

For each experiment, we used the following procedure [15]:

1. **Data preparation**: To determine the impact of data size, we selected data samples of various sizes. To do this, we developed a simple *Data-splitter* script that extracts a dataset of a specified size (in *MBs*) and splits it into a specified number of files.
2. **Keyword selection**: For each experiment, we chose one frequently occurring word (present in $\sim 20\%$ of the files), one word occurring with moderate frequency ($\sim 2\%$), and one rare word ($\sim 0.2\%$).
3. **Logical data distribution**: We created a local logical structure that described the list of Solid servers and pods subject to each experiment, with records indicating the destination of each data file and indexing files on those pods.
4. **Access control**: To simulate access control information typical of real Solid servers, we created 5000 unique WebIDs. For each text file in the pods, we granted access to a random sample of 10 of these WebIDs. We also created

[13] https://statmt.org/wmt11/translation-task.html
[14] https://data.statmt.org/news-crawl/
[15] More details on experiments setup can be found in our mentioned github repo.

four special WebIDs and granted them access to specified percentages (10%, 25%, 50%, or 100%) of files per pod.
5. **Indexing**: We created the pod indexes according to the new indexing scheme. The files and indexes for each pod are zipped and stored in a directory corresponding to the Solid server.
6. **Solid pods and MetaIndexes creation** We created the specified pods on the specified servers and put MetaIndexes in the *ESPRESSO Pod* created on each server (see Fig. 1).
7. **Data and index deployment**: The zip files are uploaded to the corresponding VMs hosting the servers and unzipped into pods according to the logical structure described above.
8. **Executing the search**: In each experiment, we executed search queries for the *three* chosen keywords. We ran each query *five* times, excluding the initial run to mitigate any *warm-up* bias, and then calculated the average of the remaining *four* run times.

Experiments & Evaluation We measured the *latency* of query searches by response time, in milliseconds. Specifically, we logged the time between sending the initial query to the overlay network and getting all the results back from all the Solid servers.

4.2 Experimental Parameters

We outline the experimental setup and configuration parameters that define the characteristics of our experimental environment and influence the performance of the system. The parameter values were chosen to represent typical real-world scenarios and were subject to the technical limitations of the experiment environment. For example, in the majority of our experiments the dataset sizes per pod allow for using approximate $5KB$-files which is the median file size found in typical user collections [8]. Some upper limits are due to disk space limits for holding both data and the indexing files (see details in Section 6).

- **Data Size per Pod (D)**: Increasing the total size of the experimental data (in text files) within each pod can increase the number and size of indexing files, which in turn may impact the overall system performance. To assess this impact, we varied the total data size per pod.
- **Number of Files per Pod (F)**: Within each user pod, splitting the data into a larger number of individual files may increase the size of the indexing files, which can also impact overall system performance. To assess this impact, we varied the number of files per pod.
- **Number of Pods per Server (P)**: The more user pods on each server, the more HTTP GET requests are generated during keyword-based search operations, directly impacting search performance. To assess this impact, we varied the number of pods per server.
- **Number of Servers (S)**: the number of Solid servers has an impact on the scalability of overlay routing and data federation across the GaianDB federated network. To assess this impact, we varied the numbers of Solid servers deployed in conjunction with GaianDB federated nodes.

– **Access Percentage ($A\%$)**: percentage of files accessible to the search party. One of the most important features of Solid is access control. To assess the impact of access control on search performance, we varied the percentages of files accessible to the search parties.

We conducted two sets of experiments. In the **first set of experiments,** we aim to see how *individually* changing, tuning, or scaling these parameters impacts the overall performance and scalability potential of the search process in the ESPRESSO system. The parameter values for the **first set of experiments** are presented in Fig. 3. The first branch shows the first group of the first set of experiments (E1–E4), which aims to evaluate the impact of changing the Data Size per Pod (D) (1MB, 5MB, 10MB, or 20MB) while keeping the other parameters fixed. In the second group of experiments (E2 and E5–E8), we evaluate the impact of changing the number of Files (F) per pod (1, 10, 50, 100, and 1000 files), while fixing the other parameters as shown in the second branch of the tree. The third group of experiments (E2 and E9–E11) evaluates the impact of changing the number of Pods (P) per Server (1, 10, 50, and 100 pods) while fixing the other parameters. The third group (E2 and E13–E15) evaluates the impact of changing the number of Solid Servers (S) (1, 10, 25, and 50 servers). The fourth group (E15 and E16-E18) checks the impact of changing the search party's Access Percentage (A%) to files in the pods (10%, 25%, 50%, and 100%)[16].

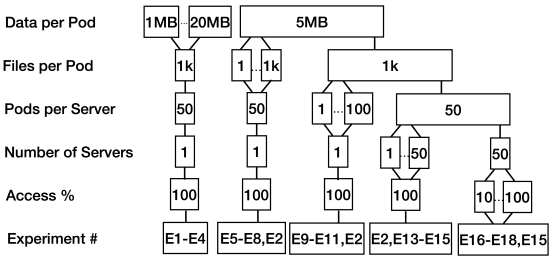


Fig. 3: Parameter values for the first set of experiments.

With the **second set of experiments,** we aim to investigate the impact of *data distribution* on search performance. To do this, we kept the total amount of data and number of files constant while changing the other parameters. The parameter values chosen for the second set of experiments are presented in Fig. 4. In the first group of the second set of experiments (E20–E24), we distribute the same amount of Total Dataset Size (50MB) among different numbers of pods (1, 5, 10, 50, and 100).

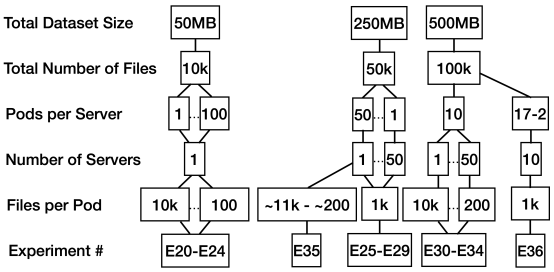


Fig. 4: Parameters for the second set of experiments.

[16] Some groups share experiments, e.g. E2, because they fit multiples series of chosen parameter values.

In the second group of experiments (E25–E29) we fix the Data per Pod (D) size (5MB) and the total number of pods (50), but distribute the pods across different numbers of servers (1, 5, 10, 25 and 50 servers). In the third group (E30–E34) we keep the number of pods per Server (P) constant (10 pods) while changing the number of servers. In the fourth group (E35, E36) we model the *non-uniform* distributions of data: in E35 files are distributed among pods according to a *power law distribution*, so the number of files per pod ranges from 200 to approximately 11,000. In experiment E36 servers are distributed among pods according to a power law distribution, so the number of pods per server ranges from 2 to 17).

Due to space limits, we keep tables of experiments exact parameter values in our mentioned GitHub repository.

5 Experimental Results and Discussion

In this section, we present and discuss the results of the experiments [17]. As evident in the results of the first set of experiments (Figure 5), the keyword's frequency had a notable impact on the search run times in all the experiments. Specifically, the more frequent the keyword, the longer it takes to retrieve the search results. In most cases, searching for the frequently occurring word, takes longer than for the word occurring with moderate frequency, which takes longer than for the rare word. This is because more frequent keywords lead to larger and more numerous search results that need to be retrieved from the Solid pod(s).

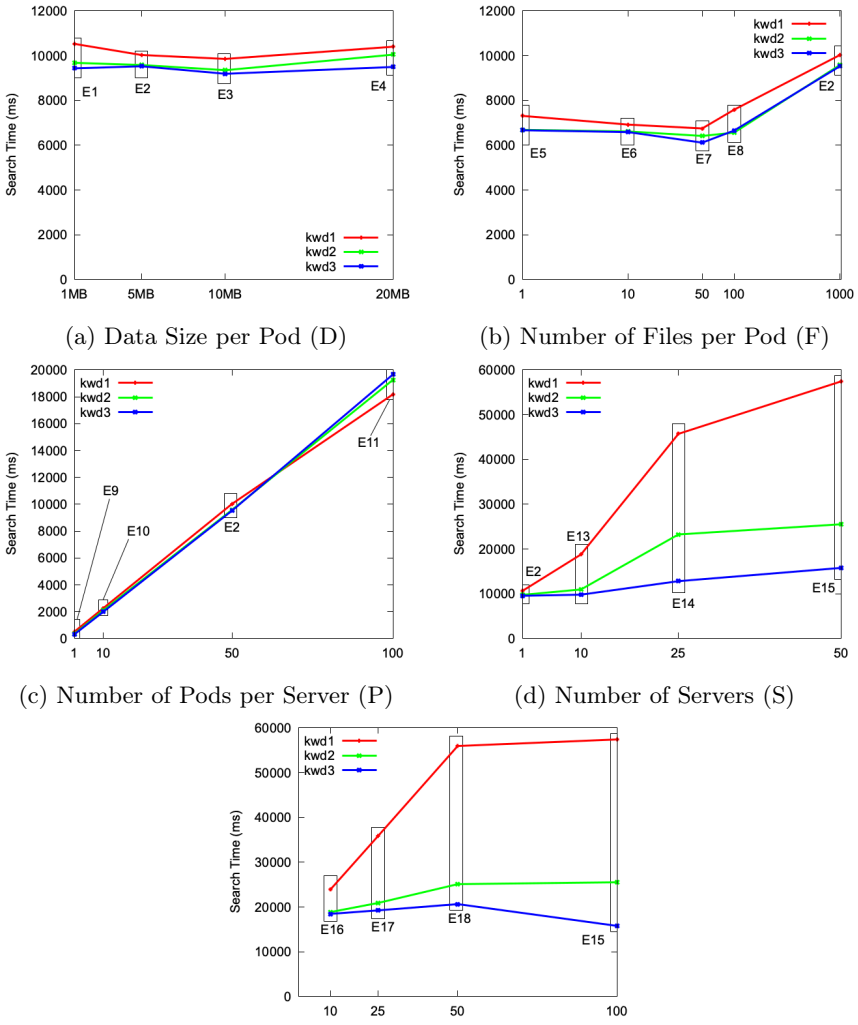
The **Data Size per Pod (D)** parameter did not have much impact on the performance (see Figure 5 (**a**)). The retrieval of index files took approximately the same time in all the experiments. The indexing scheme proposed in this paper (Section 3.3) makes the performance of keyword search *independent* of the data size in the pod, as it creates indexing files for each unique word in the indexed files. However, even though the Data Size per Pod (D) would impact the number of indexing files for each pod, our *trie-like* index structure (Section 3.3) keeps the number of files in each sub-directory sufficiently low.

The **Number of Files per Pod (F)** parameter exhibits a marked influence on search performance above a threshold of 100 files per pod, as illustrated in Figure 5 (**b**). The notable increase above this threshold can be attributed to the larger volume of search results, requiring more time to retrieve. Additionally, the size of the keyword indexing files (.ndx) increases with the number of files within a pod, further contributing to longer search response times.

The search performance is most influenced by the **Number of Pods per Server (P)** parameter. This critical impact stems from the limitations of the current Pod Searching App (CoffeeFilter), which lacks support for parallel processing, as detailed in the Limitations and Challenges Section 6. Due to this constraint, the CoffeeFilter App searches through the pods on a server sequentially. As a result, as shown in Figure 5 (**c**), search time increases linearly with the number of pods per server.

The **Number of Servers (S)** parameter markedly influences search performance in two ways: (1) it brings a greater volume of search results, where

[17] Full experiments & results can be seen in our repo: `https://shorturl.at/bgX38`



(a) Data Size per Pod (D)

(b) Number of Files per Pod (F)

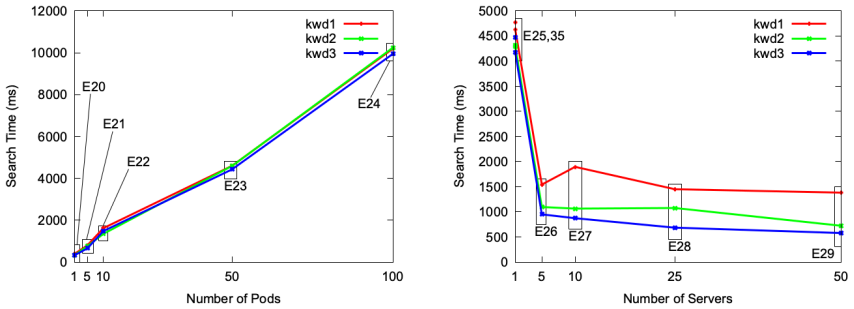
(c) Number of Pods per Server (P)

(d) Number of Servers (S)

(e) Access Percentage (A%)

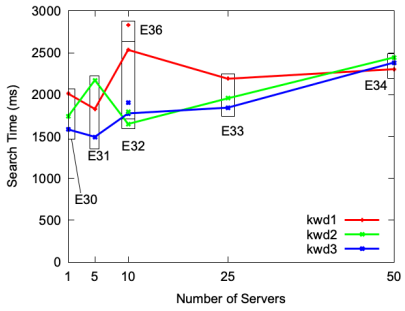
Fig. 5: The search results of the first set of experiments. Search Time in ms.

the *CoffeeFilter* Pod Searching App gathers data from many Solid servers, each hosting many pods. Additionally, (2) the routing, propagation of queries, and aggregation of results across the GaianDB overlay network also impact the overall search response time. Indeed, as illustrated in Figure 5 (**d**), there is a proportional increase in both search and routing time as the number of Solid servers increases. We also observe that as the number of servers in the network increases, the search app's performance can be influenced by the slowest server in the overlay network when fetching data from that server's Solid pods.



(a) Same Dataset size over Different Number of Pods (P)

(b) Same Number of Pods over Different Number of Servers



(c) Same Number of Pods per Server

Fig. 6: The results of the second set of experiments. Search Time in ms.

The **Access Percentage** ($A\%$) parameter affects performance in a similar way to keyword frequency. In essence, search is faster when fewer files are pertinent. As shown in Figure 5 (**e**), the search system retrieves results more slowly when the user has greater access to files within the Solid pods. Besides this performance observation in Figure 5 (**e**), it is also important to mention that the decentralised search in ESPRESSO is indeed able to preserve the privacy of users' personal data when accessed by different users/applications with varying access rights.

The results of the second set of experiments (Figure 6), exploring different distributions for the same data, again show that the parameter with the most impact on the search performance is the Number of Pods (P) on the same Solid server (Fig. 6 (**a,b**)). Search is fastest when all of the data is in the same pod (Fig. 6 (**a**)), confirming the intuition that data centralisation would lead to faster search. Fig. 6 (**b**) also shows that, for a large number of pods (e.g. 50), having them all on the same Solid server (e.g., P=50 and S=1) leads to noticeably longer search times than splitting the same number of pods over many servers (e.g., P=1 and S=50). With a constant number of pods per server, the search time grows slightly with the number of servers, but the amount of data in each pod has almost no effect (Fig. 6 (**c**)). The non-uniform distributions of data

(E35, E36), again, produce intuitive results: it makes almost no difference how we distribute data across pods (Fig. 6 (**b**)), but when we distribute the pods across the servers, search time scales with the largest number of pods on one server (Fig. 6 (**c**)).

6 Limitations & Challenges Ahead

There are some challenges to overcome and improvements to the ESPRESSO system to be made.

Parallel processing: First, the current implementation of the Pod Searching App (CoffeeFilter) does not support parallel HTTP GET requests to the indexing files in the pods because *Community Solid Server* (CSS) currently does not support multi-threading on our VM hardware specifications. Therefore, after obtaining the pod index addresses from the MetaIndex, the individual indexing file requests sent for keyword searches are processed sequentially across pods in the server - taking longer for combinations of multiple servers with multiple pods each. Enabling multi-threading on the servers and allowing CoffeeFilter to search pods in parallel should improve performance.

Routing and query propagation: Queries are currently propagated through the GaianDB overlay network by the default GaianDB *query flooding* propagation technique [3]. We believe that implementing more efficient routing algorithms and query propagation techniques over the GaianDB federated network will further enhance the search performance in ESPRESSO. Designing and maintaining additional metadata can also improve performance by reducing the stress on the overlay network and routing the queries directly to the relevant federated nodes.

Solid Servers Deployment and Latency In this paper, the experimental setup did not diversify the physical locations of deployed VMs, potentially affecting network latency results. The future investigations will aim to simulate a more realistic network environment by distributing VMs across various cloud datacenters or with different cloud providers.

Index size: Due to the index structure, which contains many small files owing to a high number of rare words (as observed in experiments E15–18 where about 90% of words occur in less than 0.002% of files), the index, although around 80–100% of the data size, occupies substantial disk space in the backend file system. This issue can be addressed by modifying the file system partitioning or enhancing the index structure. An ideal new index structure should be more compact while preserving the current structure's exposure-limiting characteristics. Additionally, our current indexing technique only tracks word frequency within documents, and adopting more sophisticated relevance measures can enhance the quality of ranked search results.

Search sophistication: Finally, an additional challenge is providing not only keyword search functionality but also allowing for more sophisticated searches: more than one word per query, *NLP* search operations, and structured queries (e.g. SPARQL) on the distributed data. This will require investigating new indexing structures for RDF data stored in Solid pods [19] as well as investigating adequate mechanisms for considering RDF data access control [22] to fit within our ESPRESSO bundle. Empowering the ESPRESSO framework with

such potentials will make it possible to do like-for-like comparisons with existing decentralised search systems such as [19] and [22].

7 Conclusion & Future Work

ESPRESSO offers search capabilities for personal online data repositories, while respecting users' access control preferences and safeguarding their data sovereignty. The system's design reduces exposure of irrelevant data in response to search queries. Our experiments affirm that ESPRESSO represents a practical and scalable solution for keyword searches, making it well-suited for a range of real-world scenarios. Furthermore, the results underscore its potential value in applications dealing with substantial volumes of sensitive data subject to stringent access control policies, such as medical data. This contrasts with traditional methods, which often require data centralisation or centralised indexing, increasing the risk of data breaches and unauthorised third-party exposure. In our future work, we plan to address the current performance and scalability challenges of the ESPRESSO system and further explore its real-world applicability in domains benefiting from decentralised, privacy-focused search and recommendation solutions [13]. This includes conducting extensive benchmarking experiments to compare centralized search baselines with ESPRESSO's decentralized approach, alongside considerations of real-world network delays. ESPRESSO holds promise for various applications, such as contact tracing where users' location data is securely stored in their pods. This approach ensures privacy while enabling notifications for users sharing locations with those testing positive for diseases like *COVID-19*.

Acknowledgements This work was funded by the UK EPSRC ESPRESSO grant (EP/W024659/1, EP/W024659/1). For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising. No new data were created during this study.

References

1. Abiteboul, S., André, B., Kaplan, D.: Managing your digital life. Communications of the ACM **58**(5), 32–35 (2015)
2. Balakrishnan, H., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Looking up data in p2p systems. Communications of the ACM **46**(2), 43–48 (2003)
3. Bent, G., Dantressangle, P., Vyvyan, D., Mowshowitz, A., Mitsou, V.: A dynamic distributed federated database. In: Proc. 2nd Ann. Conf. International Technology Alliance (2008)
4. Berners-Lee, T.: Long live the Web. Scientific American **303**(6), 80–85 (2010)
5. Chen, Y., Wang, W., Liu, Z., Lin, X.: Keyword search on structured and semi-structured data. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. pp. 1005–1010 (2009)
6. Crestani, F., Markov, I.: Distributed information retrieval and applications. In: Advances in Information Retrieval: 35th European Conference on IR Research, ECIR 2013, Moscow, Russia, March 24-27, 2013. Proceedings 35. Springer (2013)
7. Dedecker, R., Slabbinck, W., Hochstenbach, P., Colpaert, P., Verborgh, R.: What's in a pod?–a knowledge graph interpretation for the solid ecosystem (2022)

8. Dinneen, J.D., Nguyen, B.X.: How big are peoples' computer files? file size distributions among user-managed collections. Proceedings of the Association for Information Science and Technology **58**(1), 425–429 (2021)
9. Hartig, O.: An overview on execution strategies for linked data queries. Datenbank-Spektrum **13**, 89–99 (2013)
10. Kahle, B.: Locking the Web open: A call for a decentralized Web. Brewster Kahle's Blog (2015)
11. Konstantinidis, G., Holt, J., Chapman, A.: Enabling personal consent in databases. Proc. VLDB Endow. **15**(2), 375–387 (oct 2021)
12. Mansour, E., Sambra, A.V., Hawke, S., Zereba, M., Capadisli, S., Ghanem, A., Aboulnaga, A., Berners-Lee, T.: A Demonstration of the Solid Platform for Social Web Applications. In: Proceedings of the 25th international conference companion on world wide web. pp. 223–226 (2016)
13. Moawad, M.R., Maher, M.M.M.Z.A., Awad, A., Sakr, S.: Minaret: A recommendation framework for scientific reviewers. In: the 22nd International Conference on Extending Database Technology (EDBT) (2019)
14. Mudgil, P., Sharma, A., Gupta, P.: An improved indexing mechanism to index web documents. In: 2013 5th International Conference and Computational Intelligence and Communication Networks (2013)
15. Nordström, E., Rohner, C., Gunningberg, P.: Haggle: Opportunistic mobile content sharing using search. Computer Communications **48**, 121–132 (2014)
16. Ragab, M., Savateev, Y., Moosaei, R., Tiropanis, T., Poulovassilis, A., Chapman, A., Roussos, G.: Espresso: A framework for empowering search on decentralized web. In: International Conference on Web Information Systems Engineering
17. Sambra, A.V., Mansour, E., Hawke, S., Zereba, M., Greco, N., Ghanem, A., Zagidulin, D., Aboulnaga, A., Berners-Lee, T.: Solid: a platform for decentralized social applications based on linked data. MIT CSAIL & Qatar Computing Research Institute, Tech. Rep. (2016)
18. Taelman, R., Van Herwegen, J., Vander Sande, M., Verborgh, R.: Comunica: a modular SPARQL query engine for the Web. In: The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part II 17. pp. 239–255. Springer (2018)
19. Taelman, R., Verborgh, R.: Link traversal query processing over decentralized environments with structural assumptions. In: International Semantic Web Conference. pp. 3–22. Springer (2023)
20. Tiropanis, T., Poulovassilis, A., Chapman, A., Roussos, G.: Search in a redecentralised web. In: Computer Science Conference Proceedings: 12th International Conference on Internet Engineering; Web Services (InWeS 2021) (December 2021)
21. Translation, S.M.: Sixth workshop on statistical machine translation (2011)
22. Vandenbrande, Maarten and Jakubowski, Maxime and Bonte, Pieter and Buelens, Bart and Ongenae, Femke and Van den Bussche, Jan: POD-QUERY: Schema Mapping and Query Rewriting for Solid Pods. p. 5 (2023)
23. Vechtomova, O.: Introduction to information retrieval christopher d. manning, prabhakar raghavan, and hinrich schütze (stanford university, yahoo! research, and university of stuttgart) cambridge: Cambridge university press, 2008, xxi+ 482 pp; hardbound, isbn 978-0-521-86571-5 (2009)